AJAX, JSON

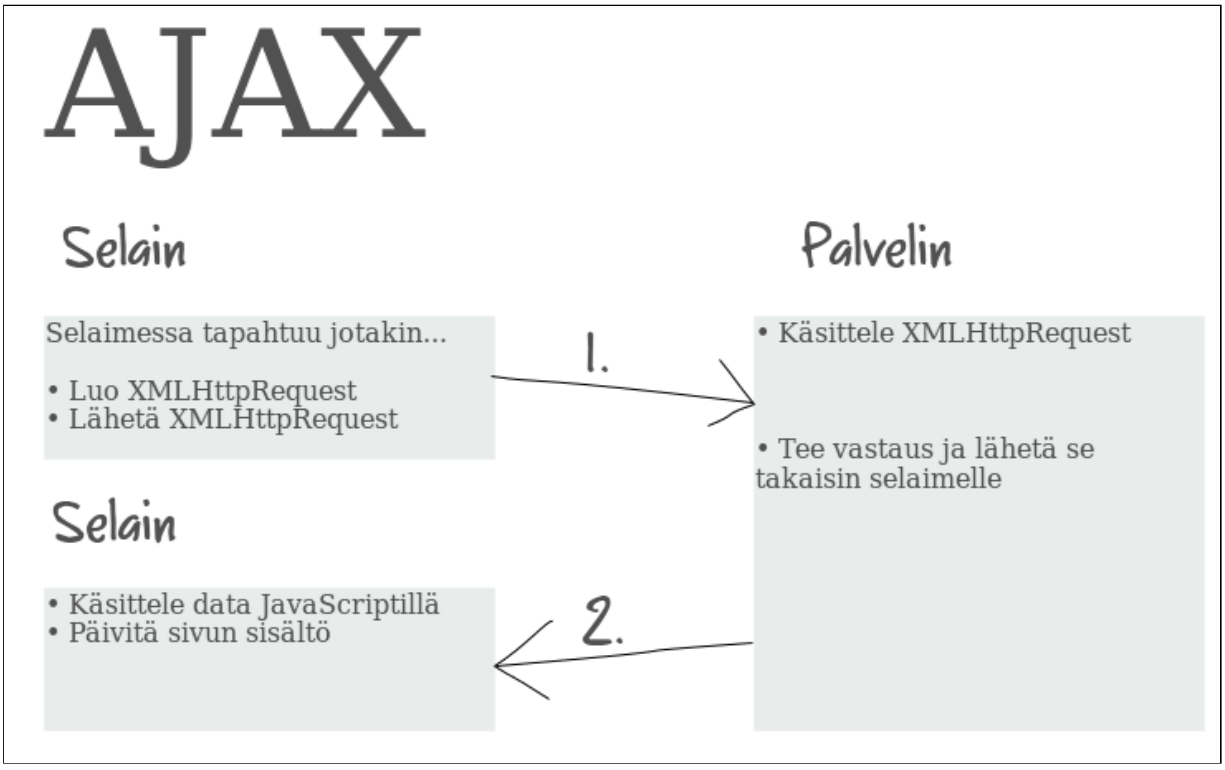
JavaScript ja AJAX

- AJAX = Asynchronous JavaScript and XML
- AJAX myös yleisnimitys usean tekniikan yhteiskäytölle: HTML, CSS, DOM, JavaScript, XML ja XMLHTTPRequest
- Perustuu **XMLHttpRequest**-olioon, joka kykenee ilman käyttäjän toimenpiteitä
  - lähettämään HTTP-pyyntöjä taustalla
  - vastaanottamaan HTTP-vastauksia taustalla
- Selaimessa sijaitsevan web-sivun lisäksi tarvitaan tavallisesti myös palvelinpuolen tekniikoita (PHP, Java, Python, jne...)

Ajaxilla voidaan

- **parantaa käyttäjäkokemusta**, koska sillä voidaan
  - muokata web-sivun rakennetta, sisältöä ja ulkoasua dynaamisesti ilman, että **koko** web-sivua pitäisi ladata uudelleen
  - saavuttaa parempi suorituskyky käyttäjän kannalta (voidaan ladata selaimeen sellaista sisältöä valmiiksi, jota käyttäjä todennäköisesti seuraavaksi tarvitsee)

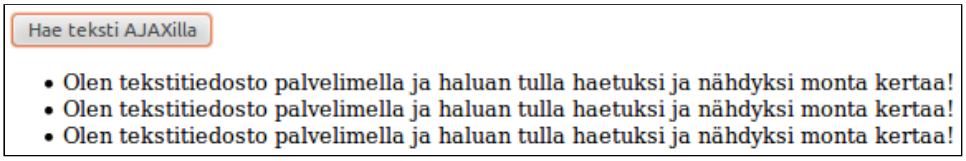
AJAX - toimintaperiaatte



- XMLHTTPRequest-pyynnöt voidaan välittää **vain samalle** web-palvelimelle mistä JavaScript-ohjelma on haettu

- Tavallisimmat dataformaatit viestinnässä palvelimen kanssa ovat XML ja JSON. (muitakin tapoja voi käyttää)
- ohjelman suoritus ei pysähdy XMLHTTPRequest-pyynnön ajaksi, vaan kutsu suoritetaan asynkronisesti taustalla
- XMLHTTPRequest-pyynnön vastauksen saavuttua suoritetaan pyynnön yhteydessä määritelty funktio
- Microsoftin vanhempia selaimia varten tarvitaan vaihtoehtoinen käsittely XMLHTTPRequest-objektille

AJAX-esimerkki 0501



`text-for-ajax.txt`

```
Olen tekstitiedosto palvelimella ja haluan tulla haetuksi ja nähdyksi monta kertaa!
```

`01-first-ajax-example.html`

```
<!DOCTYPE html>
<html>
<!-- 01-first-ajax-example.html -->
<head>
<script>
function ajaxRequest(url, funktio) {
   var xmlhttp;
   if (window.XMLHttpRequest) {
     xmlhttp=new XMLHttpRequest();
   } else {// IE5, IE6
     xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
   }

   xmlhttp.onreadystatechange=function() {
     if (xmlhttp.readyState==4 && xmlhttp.status==200)
      funktio(xmlhttp.responseText);
   }

   xmlhttp.open("GET",url,true); //true = asynkronointi päällä
   xmlhttp.send(); //GET-metodia käytettäessä ei välitetä mitään
}

function fetchAndAppend2List() {
  // Huomaa 2. parametrina funktio
  ajaxRequest("text-for-ajax.txt", function(response) {
    var element = document.getElementById("mylist");
    var li = document.createElement("li");
    var text = document.createTextNode(response);
    li.appendChild(text);
    element.appendChild(li);
  });
}
</script>

</head>

<body>
<button type=button onclick="fetchAndAppend2List();">
  Hae teksti AJAXilla
</button>
<br>
<ul id="mylist">
</ul>
```

```
</body>
</html>
```

1 Ajax-tapahtuman käynnistys

```
<button type=button onclick="fetchAndAppend2List();">
  Hae teksti AJAXilla
</button
```

2 fetchAndAppend2List

- kutsuu ajaxRequest-funktiota lisätäkseen myöhemmin siltä saamansa vastauksen (response) UL-listaan

```
ajaxRequest("text-for-ajax.txt", function(response) {
...
```

3 ajaxRequest-funktio

- luo XMLHttpRequest-objektin
- asettaa tapahtumakäsittelijäfunktion (nimeltä `funktio`) [onreadystatechange]
- avaa yhteyden palvelimelle ja lähettää pyynnön

4 ajaxRequest-funktio jatkaa

- Palvelin palauttaa vastauksen
- XMLHttpRequest-objekti kutsuu tapahtumakäsittelijäfunktiota (nimeltä `funktio`)

5. fetchAndAppend2List

lisää **tekstinä** (xmlhttp.responseText) saamansa vastauksen UL-listaan

XMLHttpRequest - Ominaisuudet ja toiminnot

- XMLHTTPRequest-objektin ominaisuudet ja toiminnot on kuvattu W3C:n The XMLHTTPRequest-Object -dokumentaatiossa

onreadystatechange	tapahtumankäsittelijä
readyState	XMLHTTPRequest-objektin tila 0 = ei vielä alustettu 1 = odottaa lataamista 2 = lähetetty ja latautuu 3 = varsinaista dataa lähetetty, prosessointi kesken 4 = valmis
responseText	palautettu data merkkijonona
responseXML	palautettu data XML-datana
status	HTTP-statuskoodi (esim. 200 on OK ja 404 Not Found, jne...)
statusText	HTTP-satuskoodiin liittyvä mahdollinen teksti, esim. virheilmoitus

toiminnot:

abort()	käynnissä olevan pyynnön keskeyttäminen
setRequestHeader("name","value")	asettaa pyynnön otsikkotietoja
getResponseHeader("name")	palauttaa halutun HTTP-otsakkeen arvon

getAllResponseHeaders()	palauttaa kaikki HTTP-otsakkeet
open("method", "URL"[, async[, "login"[, "passwd"]]])	asettaa metodin GET/POST/jne..., osoitteen, mahdollisen asynkronoinnin, käyttäjän ja salasanan
send(data)	lähettää pyynnön Web-palvelimelle ja mahdollisen datan

Datan käsittely ja jäsentäminen XMLHTTPRequest-objektista

a) responseText

- tallentaa välitetyn datan merkkijonona
- data tulee käsitellä merkkijonosta
- voi sisältää myös JSON-muotoista dataa (palataan myöhemmin)

b) responseXML

- tallentaa välitetyn datan DOM-rakenteen olioina
- data voidaan käydä läpi DOM-metodeilla ja ominaisuuksilla (getElementsByTagName(), childNodes[], parentNode, jne...)

JavaScript DOM ominaisuuksia ja metodeja (yleisimmin tarvittavat):

childNodes	lapsisolmut taulukossa
firstChild	ensimmäinen lapsisolmu
lastChild	viimeinen lapsisolmu
nodeName	solmun nimi
nodeType	solmun tyyppi
nodeValue	solmun arvo
nextSibling	seuraava sisarsolmu (yhteinen vanhempi)
previousSibling	edellinen sisarsolmu (yhteinen vanhempi)
parentNode	solmun vanhempisolmu
appendChild	luo uuden lapsisolmun
hasChildNodes	true, jos solmulla on lapsisolmuja
removeChild	tuhoaa lapsisolmun
createAttribute	luo elementille uuden attribuutin
createElement	luo uuden elementin
createTextNode	luo uuden tekstisolmun
getElementsByTagName	luo taulukon elementeistä
getElementsById	etsii elementin ID:n perusteella

Esimerkki:

```
<?xml version="1.0"?>
<user>
   <name>Pelle</name>
   <email>pelle@pelle.com</email>
</user>
```

Tähän dataan päästään käsiksi esim. seuraavasti:

```
var nameSolmu = httpPyynto.responseXML.getElementsByTagName("name")[0];
var name = nameSolmu.childNodes[0].nodeValue;
```

JavaScript - JSON

- JSON (*JavaScript Object Notation*) on dataformaatti erityisesti tiedonvälitykseen
- Suosittu formaatti erityisesti APIen rakentamisessa
- JSON on JavaScriptin kanssa suoraviivaisempi kuin XML
- JSON ja XML omaavat paljon samanlaisia tavoitteita. On vaikea perustella kummankaan ylivertaisuutta. Googlaamalla "JSON vs. XML comparison pros and cons" löytää paljon värittynyttä keskustelua.
- JSON JavaScript-riippumaton, sitä voi käyttää muidenkin kielien kanssa

Tällaiselta se JSON näyttää

```
{
"friends":
 [
  {"firstname":"Mari", "lastname":"Ranta", "email":"mari@r.com"},
  {"firstname":"Kari", "lastname":"Ranta", "email":"kari@r.com"},
  {"firstname":"Sari", "lastname":"Ranta", "email":"sari@r.com"}
 ]
}
```

JSON-perusesimerkki 0502

Näytetään UL-listassa oheinen JSON-merkattu data:

```
<script>
var JSONObject = {"friends":[
  {"firstname":"Mari", "lastname":"Ranta", "email":"mari@r.com"},
  {"firstname":"Kari", "lastname":"Ranta", "email":"kari@r.com"},
  {"firstname":"Sari", "lastname":"Ranta", "email":"sari@r.com"}
]}

function init() {
   for(i=0; i< JSONObject.friends.length;i++) {
      var element = document.getElementById("mylist");
      var li = document.createElement("li");
      var text = document.createTextNode(
         JSONObject.friends[i].firstname+ " " +
         JSONObject.friends[i].lastname+ " " +
         JSONObject.friends[i].email);
      li.appendChild(text);
      element.appendChild(li);
   }
}

</script>

<body onload="init();">
   <ul id="mylist">
   </ul>
</body>
```

Lopputulos näyttää seuraavalta:

- Mari Ranta mari@r.com
- Kari Ranta kari@r.com
- Sari Ranta sari@r.com

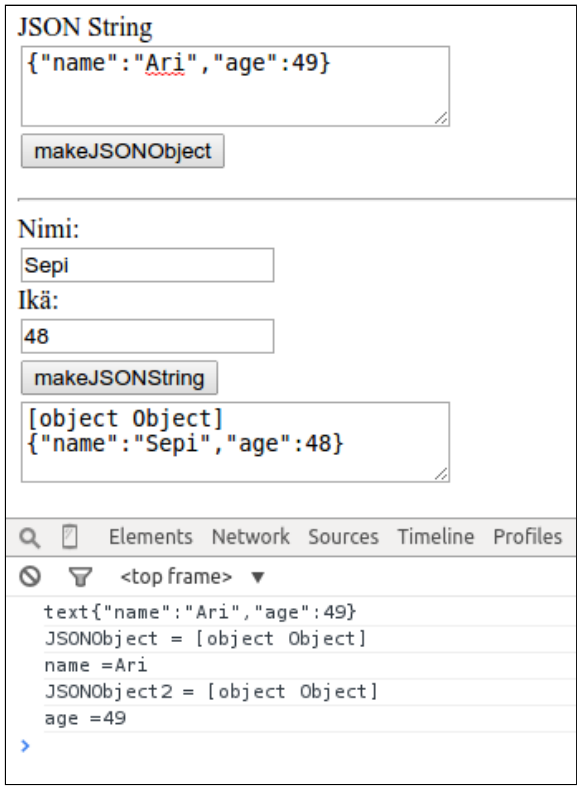
Muunnokset - JSONObject <-> JSONText

JSONin kanssa toimiessa on usein tarvetta muuntaa tekstimuotoinen JSON-data JavaScript-olioksi ja toisinpäin.

- Teksti -> Object: `JSON.parse()` tai `eval()`
- Object -> Teksti: `JSON.stringify`

Esimerkki 0503

- makeJSONObject kirjoittaa konsoliin
- makeJSONString kirjoittaa alempaan textarea-elementtiin



Lähdekoodi:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<script>
//{"name":"Ari","age":49}
function makeJSONString() {
	var name = document.getElementById("name").value;
	var age = parseInt(document.getElementById("age").value);
	var JSONObject = {
	"name":name,
	"age":age
	};
	var myJSONText = JSON.stringify(JSONObject);
	document.getElementById("textarea2").value = JSONObject
	  + "\n" + myJSONText;
}

function makeJSONObject() {
	var text = document.getElementById("textarea1").value;
	console.log("text" + text);

	// eval:
```

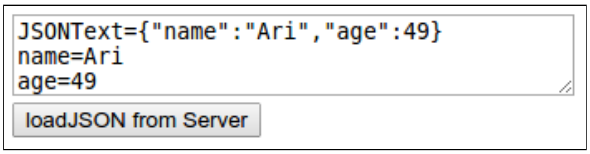
```
        var JSONObject = eval("("+ text + ")");
        console.log("JSONObject = "+JSONObject);
        console.log("name =" + JSONObject.name);

        //parse:
        var JSONObject2 = JSON.parse(text);
        console.log("JSONObject2 = "+JSONObject2);
        console.log("age =" + JSONObject2.age);
}

</script>

<body>
<form>
JSON String<br>
<textarea id="textarea1" cols=30 rows=3></textarea><br>
<input type=button value="makeJSONObject" onclick="makeJSONObject();">
<p></p>
<hr>
Nimi:<br>
<input type=text id=name><br>
Ikä:<br>
<input type=text id=age><br>

<input type=button value="makeJSONString" onclick="makeJSONString();">
<br>
<textarea id="textarea2" cols=30 rows=3></textarea><br>
</form>


</body>
```

JSON-datan lataamisesta AJAXilla

- Ajaxin rajoitus "XMLHTTPRequest-pyynnöt voidaan välittää **vain samalle** web-palvelimelle mistä JavaScript-ohjelma on haettu" koskee myös JSONia, muualta JSON-dataa voidaan hakea seuraavilla ratkaisuilla:
  - Omalle palvelimelle voidaan luoda resurssi (esim. PHP-ohjelmisto), joka hakee dataa muualtakin (cross domain requests)
  - CORS-mekanismin <http://en.wikipedia.org/wiki/Cross-origin_resource_sharing> avulla AJAX-pyynnöt voidaan sallia myös muualta (Jos omalle palvelimelle on riittävät oikeudet)
  - JSONP <http://en.wikipedia.org/wiki/JSONP> saattaa myös tulla kyseeseen, jos riittäviä oikeuksia omalle palvelimelle ei ole (palvelimen pitää tukea). Ei tarvi AJAXia.

JSON/AJAX-perusesimerkki 0504

Haetaan `server.php`-skriptin tulostama JSON-data `{"name":"Ari","age":49}` AJAXilla ja jäsennetään se näytettäväksi:



04-json-ajax-basic.html

```
<!-- 04-json-ajax-basic.html -->
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<script>
function ajax(url, fn) {
        try {
                var xhr = new XMLHttpRequest();
        } catch(e) {
                alert(e);
        }
```

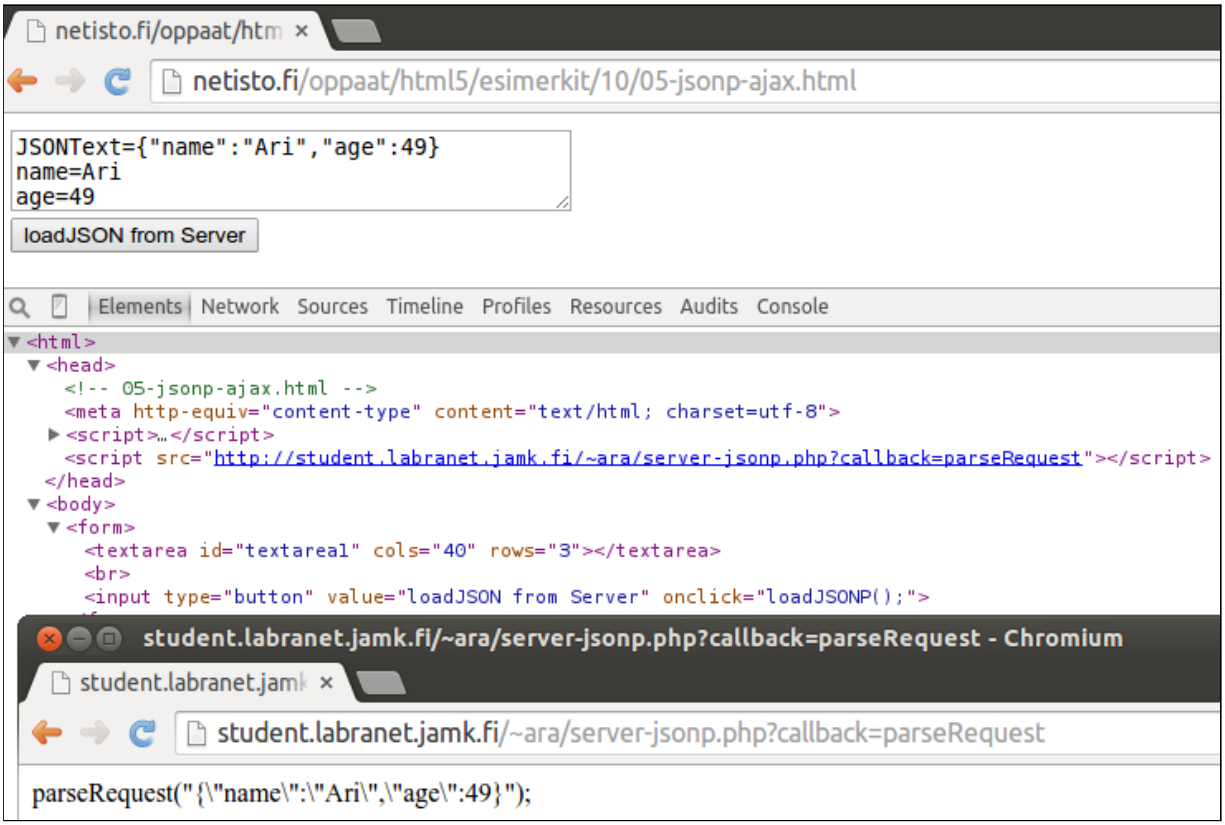
```
        xhr.open("GET", url, true);
        xhr.send();
        xhr.onreadystatechange = function() {
        if(xhr.readyState == 4 && xhr.status==200) {
           fn(xhr.responseText);
      }
   }
}

function loadJSON() {
   ajax("server.php", function(response) {
      var JSONObject = JSON.parse(response);
      document.getElementById("textarea1").value =
      "JSONText="+response+"\nname=" + JSONObject.name
      + "\nage=" + JSONObject.age;
   });
}
</script>

<body>
<form>
<textarea id="textarea1" cols=40 rows=3></textarea><br>
<input type=button value="loadJSON from Server" onclick="loadJSON();">
</form>


</body>
```

server.php

```
<?php
// server.php
$json_string = '{"name":"Ari","age":49}';
echo $json_string;
?>
```

JSONP - Datan lataaminen eri palvelimelta

- JSONP ~ "JSON with padding"
  - "Padding" on tuo funktion nimi JSON-datan edessä
- HTML-dokumentti JavaScripteineen netisto.fi-palvelimella
  - luo script-elementin, johon hakee PHP-ohjelmalta suoritettavan funktiokutsun ja JSON-datan
- PHP-skripti student.labranet.jamk.fi-palvelimella
  - tulostaa HTML-dokumentissa sijaitsevan funktiokutsun ja sille välitettävän JSON-datan.



05-jsonp-ajax.html

```
<head>
<!-- 05-jsonp-ajax.html -->
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<script>

function loadJSONP() {
   var url = "http://student.labranet.jamk.fi/~ara/server-jsonp.php?callback=parseRequest";
   var script = document.createElement('script');
   script.setAttribute('src', url);
   document.getElementsByTagName('head')[0].appendChild(script);
   /* Tämän tuloksena:
   <script src='http://student.labranet.jamk.fi/~ara/server-jsonp.php?callback=parseRequest' /script

   Ja skriptin sisältönä rivi:

   parseRequest({"name":"Ari","age":49});

   */
}


function parseRequest(response) {
      var JSONObject = JSON.parse(response);
      document.getElementById("textarea1").value =
      "JSONText="+response+"\nname=" + JSONObject.name
      + "\nage=" + JSONObject.age;
}
</script>

</head>
<body>
<form>
<textarea id="textarea1" cols=40 rows=3></textarea><br>
<input type=button value="loadJSON from Server" onclick="loadJSONP();">
</form>


</body>
```

server-jsonp.php

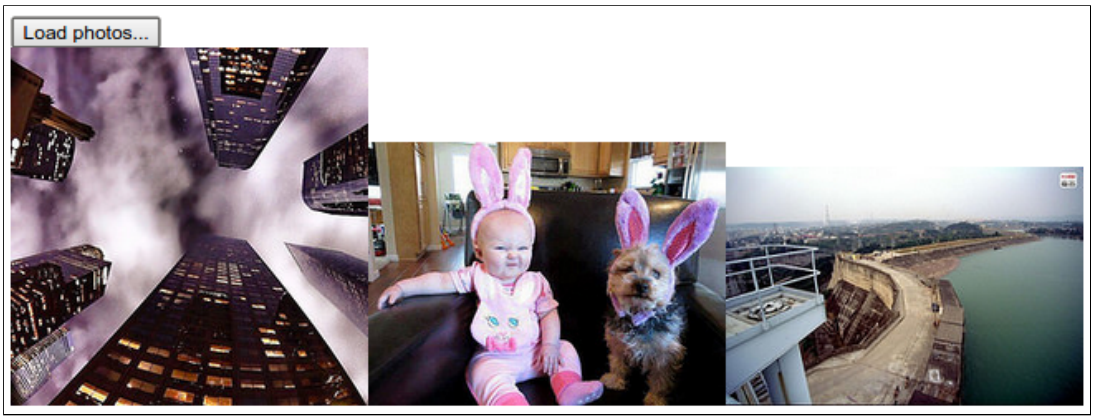
```php
<?php
// server-jsonp.php
$callback = "";
if  (isset($_GET['callback'])) {
  $callback  =  filter_var($_GET['callback'],  FILTER_SANITIZE_STRING);
}

$json_string = '{"name":"Ari","age":49}';
echo  $callback  .  '('. json_encode($json_string) . ');';

?>
```

JSONP-esimerkki: Kuvien lataaminen Flickr-palvelusta

Verkossa on runsaasti dataa tarjolla monenlaisiin käyttötarkoituksiin erilaisissa formaateissa. Myös JSONia on tarjolla runsaasti

- http://apisuomi.fi/rajapinnat-kompaktisti/
- http://www.flickr.com/services/api/
- http://dev.twitter.com
- jne.

Esimerkiksi Flickr-palvelusta voi ladata JSNOPilla julkisesti tarjolla olevia kuvia:



Ohjelmakoodi: 06-jsonp-flickr.html

```
<!doctype html>
<!-- 06-jsonp-flickr.html -->
<html lang="en">
<head>
<meta charset="UTF-8">
<title>JSONP Example - Load photos from the Flickr Server</title>
<script>

function load() {
        var script = document.createElement('script');
   // https://www.flickr.com/services/feeds/docs/photos_public/
        script.src = 'http://api.flickr.com/services/feeds/photos_public.gne?format=json&jsoncallbac
        document.getElementsByTagName('head')[0].appendChild(script);
}

function parseRequest(response) {
   // Näillä kommentoiduilla voi ensin tutkia, millaista dataa on saatavilla:
        //console.log(response);
        //console.log(response.items.length);
   var maxitems = 3;
        var items = response.items.length;

   // Näytetään korkeintaan 3 ekaa kuvaa:
   if (maxitems < items) items = maxitems;

        for (i=0;i<items;i++) {
```

```
                addImage(response.items[i].media.m);
        }
}

function addImage(src) {
        var img = document.createElement("img");
        img.setAttribute("src",src);
        document.body.appendChild(img);
}

</script>
</head>

<body>
    <button onclick="load()">Load photos...</button><br/>
</body>
</html>
```

JSONP - Riskejä?

- JSONP lisää script-elementin sivullesi
- Script-elementin sisällön määrää kolmas osapuoli, joka voi injektoida sivullesi haittakoodia:

```
/* haittakoodia */ CallBackFunction(/*JSON-data*/)
```

=> Lataa ainoastaan dataa, jonka lähteeseen luotat

Linkkejä

JSON @ W3CShools

Jätetty tarkoituksella tyhjäksi.